NISTIR 4873

Automatic Indexing

Donna Harman

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Computer Systems Laboratory
Advanced Systems Division
Image Recognition Group
Gaithersburg, MD 20899

NIST

NISTIR 4873

Automatic Indexing

Donna Harman

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards and Technology
Computer Systems Laboratory
Advanced Systems Division
Image Recognition Group
Gaithersburg, MD 20899

July 1992

U.S. DEPARTMENT OF COMMERCE
Barbara Hackman Franklin, Secretary

TECHNOLOGY ADMINISTRATION
Robert M. White, Under Secretary for Technology

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
John W. Lyons, Director

Automatic Indexing

Donna Harman
National Institute of Standards and Technology

1. Introduction

Vast amounts of text are available online today, including text created for electronic access and text designed mainly for traditional publishing. This text is not searchable without the ability to do automatic indexing. Yet the "discovery" that adequate indexing could be done using single terms from the text generally surprised the library community. As Cyril Cleverdon reported from the Cranfield project (Cleverdon & Keen 1966):

> "Quite the most astonishing and seemingly inexplicable conclusion that arises from the project is that the single term indexing languages are superior to any other type...unless one is prepared to say that the whole test conception is so much at fault that the results are completely distorted, there is no course except to attempt to explain the results which seem to offend against every canon on which we were trained as librarians."

Today we not only accept these results, but base many of the large commercial online systems on this once-revolutionary idea. The discovery of automatic indexing coincided with the availability of large computers and created a major interest in automatically indexing and searching text, such as the work done by H.P Luhn (1957) in investigating the use of frequency weights in automatic indexing. Work has continued since then in various research laboratories and has resulted in more sophisticated automatic indexing methods, using single terms and using larger chunks of text (such as phrases).

This paper was written to serve two separate goals. The first goal is to provide a tutorial on single term indexing of "real-world" text. Therefore section 2 steps through the indexing process discussing the types of critical issues that must be resolved during full text indexing in order to provide effective retrieval performance. Most of these issues are straight-forward. However poor choices of indexing parameters produce systems that would be considered failures in most applications.

The second goal is to provide some discussion of advances in automatic indexing beyond the simple single-term indexing done in most operational retrieval systems. Section 3 discusses many of the techniques being investigated and provides references for further reading.

2. Automatically producing simple index terms

This section presents a walk-through of the processing of an online text file to produce a list of index terms that can be used for searching that file. These terms would be placed in an inverted file, or other data structure, and an information search could be made against this index using Boolean retrieval operators to combine the terms. Alternatively some of the more advanced searching methods could use these terms as input to term weighting algorithms that produce ranked output using statistical techniques.

2.1 What constitutes a record

The first key decision for any indexing is the choice of record boundaries which identify a searchable unit. A record could be defined as an entire book, a chapter in the book, a section in that chapter, or even a paragraph. This decision is critical for effective retrieval, both in the retrieval/display stage and in the search stage. Often this decision is clearcut. For example if the application is searching bibliographic records as in an online catalog, clearly a record is one of the bibliographic records. Similarly, if the application is searching newspaper articles or newswire stories for particular events, then these articles or stories each becomes a record. The choice of record size becomes fuzzy, however, as the size of the documents being examined grows larger. If the documents being searched are long articles such as legal transcriptions of court cases or full journal articles, then the record might still be the entire document, although this may make display and searching more difficult. However, if the documents being searched are manuals or textbooks, a record should not be the entire document. Here the choice should depend on the retrieval and display mechanisms of the particular application. For

example the application of searching an online manual might have a record defined as the lowest subsection, so that users find and display very exact subsections of material. If the application is to provide pointers into paper copies of long articles (such as 100-page+ court cases), it might be reasonable to make each page or small section a record so that the display could show a one-line sentence with the hits, and give the page number.

The choice of record size is not only important for display, but also is critical for effective searching. A record which is too short provides little text for the searching algorithms to use and will cause poor results. Too large a record, however, may dilute the importance of word matches, and cause many false matches. For these reasons it would not be reasonable to make a sentence a record, but paragraphs might be fine as records. Alternatively it would not be effective to make a very long section a record; it would be better to break it into smaller subsections. Further, the choice of record size may also affect the choice of term weighting and retrieval algorithms (see section 3.1 on term weighting).

A recent paper (Harman & Candela 1990) shows some possible record size decisions and their consequences. Three different text collections were involved in user testing of a retrieval system using automatic indexing and statistical ranking. The first text collection was small (1.6 megabytes) and consisted of a manual organized into sections and chapters. A record was determined to be equivalent to a paragraph in this manual, because this appeared to be the most useful record size for the end users. This decision caused many short records (see Table 1). The second text collection was a legal code book, with sections and subsections. Here the records were set to be each subsection, again based on user preference. The records were therefore much larger, with many words occurring multiple times within each record. The third text collection consisted of about 40,000 court cases. A record here was set to be a court case. Table 1 shows some basic statistics on these text collections. The average number of terms per record includes duplicate terms and is a measure of the record length rather than the number of unique term occurrences. The average number of postings per term is the average number of documents containing that term.

TABLE 1
Collection Statistics

Size of collection	1.6 MB	50 MB	806 MB
Number of records	2653	6652	38304
Average number of terms per record	96	1124	3264
Number of unique terms	5123	25129	243470
Average postings per term	14	40	88

2.2 What constitutes a word and what "words" to index

The second key decision for any indexing is the choice of what constitutes a word and then which of these words to index. In manual indexing systems this choice is easily made by the human indexer. However for automatic indexing it is necessary to define what punctuation should be used as word separators and to define what "words" to index.

Normally word separators include all white spaces and all punctuation. However there are many exceptions to this rule, and, depending on the application and the searching software, the methods of handling these exceptions can be crucial to successful retrieval. The following examples illustrate some of the problems encountered in typical applications.

- Hyphens -- some words can appear in both hyphenated and unhyphenated versions. Sometimes the treatment of hyphens is critical to retrieval, such as in chemical names and other normally hyphenated elements (glycol-sebacic, F-15, MS-DOS, etc.).
- Periods -- periods can appear as a part of a word, such as computer file names (paper.version1), subsection titles (1.367A), and in company names.
- Slashes, parentheses, underscores -- these can appear as parts of words (OS/2), as parts of section titles (367(A)), and as parts of terms in programming languages (doc_no).

- Commas -- if numbers are indexed, commas and decimal points become important.

Once word boundaries are defined, an equally difficult issue is what words or tokens to index. This particularly applies to the indexing of numbers. If numbers are indexed, the number of unique words can explode because there is an unlimited set of unique numbers. As an example, when all numbers in the 50 megabyte text collection shown in Table 1 were indexed, the number of unique terms went from 10122 (indexing no numbers) to 55486 (indexing all numbers). (The number of unique terms shown in Table 1 includes indexing some numbers as explained later). Indexing all numbers would have caused an almost doubling of the index size, and therefore slower response times. However, not indexing the numbers can lead to major searching problems when a number is critical to the query (such as "what were the major breakthroughs in computer speed in 1986").

The same problem can apply to the indexing of single characters (other than the words "a" or "i", which are discussed in the next section as stopwords). Whereas the number of unique single characters are limited, the heavy use of single characters as initials, section labels, etc. can increase the size of the index. Again, however, not indexing single characters can lead to searching problems for queries in which these characters are critical (such as "sources of vitamin C").

The solutions to both the problem of word boundaries and what words to index involve compromises. Before indexing is started, samples of the text to be indexed, and samples of the types of queries to be run, need to be closely examined. This may require a prototype/user testing operation, or may be solved by simply discussing the problem with the users. The following examples illustrate some of the possible compromises.

- The punctuation in the text should be studied, and potential problems identified so that reasonable rules of word separation can be found. Often hyphenated words are treated both as separated words and as hyphenated words. Other types of punctuation are handled differently based on preceding or succeeding characters or spaces.

- The use of upper and lower case letters also needs to be determined. Usually upper-case letters are changed to lower case during indexing as capitalized words indicating sentence beginnings will not correctly match lower case query words. However, if proper nouns are to be treated as special terms, then upper-case letters are necessary for proper noun recognition.

- The indexing of numbers is also heavily application dependent. Dates, section labels, and numbers combined with alphabetics may be indexed, and other numbers not indexed. If hyphens can be kept, then some number problems are eliminated (such as F-15). In the 50-megabyte text collection shown in Table 1, numbers that were part of section labels were kept, and these were distinguished by the punctuation that appeared in the number. Some searches were still unsuccessful, however, because of the lack of complete number indexing.

- The indexing of single characters is somewhat easier to handle. Users can check the alphabet and note any letters that have particular meaning in their application, and these letters can be indexed.

Most commercial systems take a conservative approach to these problems. For example, Chemical Abstracts Service, ORBIT Search Service, and Mead Data Central's LEXIS/NEXIS systems all recognize numbers and words containing digits as index terms, and all are case insensitive. In general they have no special provisions for punctuation marks, although Chemical Abstracts Service keeps hyphenated words as single tokens, and the other two systems break hyphenated words apart (Fox 1992).

2.3 Use of stop lists

Additionally most automatic indexing techniques work with a stop list that prevents certain high-frequency or "fluff" words from being indexed. Francis & Kucera (1982) found that the ten most frequently used words in the English language typically account for twenty to thirty percent of the terms in a document. These terms use large amounts of index storage and cause poor matches (although this is not usually a problem because of the use of multiple query terms for matching purposes).

One commonly-used approach to building a stop list is to use one of the many lists generated in the past. Francis & Kucera (1982) produced a stop list of 425 words derived from the Brown corpus, and a list of 250 stopwords was published by van Rijsbergen (1975). These lists contain many of the words that always have a high frequency, such as "a", "and", "the", and "is", but also may contain "fluff" words that may not have a high

frequency for some text collections, such as "below", "near", "always", and "that". Note that unlike high frequency words, "fluff" words do not necessarily hurt retrieval performance, and will not seriously affect storage. Often these words become crucial to retrieval, such as in a query "stocks with costs below X dollars", or "restaurants near the harbor".

A more suitable method of constructing a stop list would be to produce a word frequency listing for the text to be indexed, and then examine each of the high frequency words. If there is no known importance of a given word in the application, then that word can be safely placed on a stop list. An example of this procedure is the work done at the National Institute of Standards and Technology (NIST) with a 250-megabyte collection of the Wall Street Journal. The top twenty-seven high-frequency words were examined, and four words were removed as possibly important ("a", "at", "from" and "to"). The remaining twenty-three words then became the stop list. This was a reduction from a previously-used stop list from the SMART project of 418 words. The shrinkage of the stop list caused an increase of about 25% in the index storage, but made available for searching an additional 395 words. This new stop list is shown as Table 2 as an illustration of an abbreviated stop list rather than as a particularly recommended one.

TABLE 2
Sample Stop Words

an	been	in	or	which
and	but	is	that	will
are	by	it	the	with
as	for	of	this	
be	have	on	was	

It should be noted that commercial systems are even more conservative in the use of stop lists. ORBIT Search Service has only eight stop words: "and", "an", "by", "from", "of", "or", "the", and "with" (Fox 1992). The MEDLARS system has even fewer stop words.

2.4 Use of suffixing or stemming

Many information retrieval systems also use suffixing or stemming to replace all indexed words with their root forms. Different stemming algorithms have been used, including "standard" algorithms, and algorithms built for a specific domain such as medical English (Pacak 1978). For a survey of the various algorithms see Frakes (1992). Three standard algorithms, an "S" stemming algorithm, the Lovins (1968) algorithm, and the Porter (1980) algorithm, are most often used, and the following excerpts (Harman 1991) show some of their characteristics.

The "S" stemming algorithm, a basic algorithm conflating singular and plural word forms, is commonly used for minimal stemming. The rules for a version of this stemmer, shown in Table 3, are only applied to words of sufficient length (three or more characters), and are applied in an order dependent manner (i.e., the first applicable rule encountered is the only one used). Each rule has three parts: a specification of the qualifying word ending, such as "ies"; a list of exceptions; and the necessary action.

TABLE 3 An "S" Stemmer
IF a word ends in "ies", but not "eies" or "aies" THEN "ies" --> "y" IF a word ends in "es", but not "aes", "ees", or "oes" THEN "es" --> "e" IF a word ends in "s", but not "us" or "ss" THEN "s" --> NULL

The Lovins stemmer works similarly, but on a much larger scale. It contains a list of over 260 possible suffixes, a large exception list, and many cleanup rules. In contrast, the Porter algorithm looks for about 60 suffixes, producing word variant conflation intermediate between a simple singular-plural technique and Lovins algorithm. Table 4 shows an example of the differences among the three stemmers. The first column shows the actual words (full words) from the query. The next three columns show the words that are conflated with the original words (words that stem to the same root for that stemmer) based on three different stemmers. The starred terms are the ones that were useful in retrieval for this particular query and are shown only to indicate the "quasi-random" matching that occurs when matching query terms with terms in relevant documents.

TABLE 4
Stemmer Differences for query 109 of the Cranfield test collection

Query -- panels subjected to aerodynamic heating

FULL WORD	S	PORTER	LOVINS
*panels	*panel *panels	*panel *panels	*panel *panels
subjected	subjected	subjected *subject subjective subjects	subjected *subject subjective subjects
*aerodynamic	*aerodynamic aerodynamics	*aerodynamic aerodynamics *aerodynamically	*aerodynamic aerodynamics *aerodynamically aerodynamicist
*heating	*heating	*heating *heated	*heating *heated *heat heats heater

Stemming or suffixing is done for two principal reasons: the reduction in index storage required and the increase in performance due to the use of word variants. The storage savings using stemming is data and implementation dependent. For small text collections on machines with little storage, a sizable amount of inverted file storage can be saved using stemming. For the 1.6 megabyte manual shown in Table 1, approximately 20% of storage was saved by using the Lovins stemmer. Lennon et. al. (1981) showed compression percentages for the Lovins stemmer of 45.8% for the Brown Corpus. However, for the larger text collections normally used in online retrieval, less storage is saved. The savings was less than 14% for the text of 50 megabytes in Table 1, probably because this text contains large amounts of numbers, misspellings, proper names, etc. (items that usually cannot be stemmed).

In terms of performance improvements, research has shown that *on the average* results were not improved by using a stemmer. However, system performance must reflect a user's expections, and the use of a stemmer (particularly the S stemmer) is intuitive to many users. The OKAPI project (Walker & Jones 1987) did extensive work on improving retrieval in online catalogs, and strongly recommended using a "weak" stemmer at all times, as the "weak" stemmer (removal of plurals, "ed" and "ing") seldom hurt performance, but provided significant improvement. They found drops in precision for some queries using a "strong" stemmer (a variation of the Porter algorithm), and therefore recommended the use of a "strong" stemmer only when no matches were found. One method of selective stemming is the availability of truncation in many online commercial retrieval systems. However, Frakes (1984) found that automatic stemming performed as well as truncation by an experienced user, and most user studies show little actual use of truncation. Given today's retrieval speed and the ability for user interaction, a realistic approach for online retrieval would be the automatic use of a stemmer, using an algorithm like Porter or Lovins, but providing the ability to keep a term from being stemmed (the inverse of truncation). If a user found that a term in the stemmed query produced too many nonrelevant documents, the query could be resubmitted with that term marked for no stemming. In this manner, users would have full advantage of stemming, but would be able to improve the results of those queries hurt by stemming.

3. Advanced automatic indexing techniques

The basic index terms produced by the methods discussed in section 2 can be used "as is", with Boolean connectors to combine terms, or a single term may be used for simple searches. However, researchers in information retrieval have been developing more complex automatic indexing techniques for over thirty years, and having varying degrees of success with these new techniques doing experiments with small test collections. Some of these techniques (such as the term weighting discussed in section 3.1) are clearly successful and are likely to scale easily into large full-text documents. Other techniques, such as the query expansion techniques described in section 3.2, do well on small test collections, but may need additional experimentation when used in large full-text collections. The added discrimination provided by using phrases as indexing terms rather than only single terms is discussed in section 3.3. In general the use of phrases has not been successful in small test collections, but is likely to become more useful, or even critical, in large full-text documents. Large full-text collections may need better term discrimination measures, and some recent experiments in selecting better indexing features or in providing more advanced term weighting are described in sections 3.4 and 3.5. Finally, the notion of combining evidence from multiple types of document indexing is presented in section 3.6.

3.1 Term weighting

Whereas terms coming from automatic indexing can be used without weights, they offer the opportunity to do automatic term weighting. This weighting is essential to all systems doing statistical or probabilistic ranking. Many of the commercial systems provide an ability to rank documents based on the number of terms matching between the query and the document, but find that users do not select this option often because of poor performance. There are several reasons for this poor performance:

1. There is no technique for resolving ties. If there are three words in a query, it may be that only a few documents match all three words, but many will match two terms, and these documents are essentially unranked with respect to each other.

2. There is no allowance for word importance within a text collection. A query such as "term weighting in information retrieval" could return a single document containing all four non-common words, and then an unranked list of documents containing the two words "term" and "weighting" or "information" and "retrieval", all in random order. This could mean that the possibly 10 documents containing "term" and "weighting" are buried in 500 documents containing "information" and "retrieval".

3. There is no allowance for word importance within a document. Looking again at the query "term weighting in information retrieval", the correct order of the documents containing "term" and "weighting" would be by frequency of "weighting" within a document, so that the highest ranked document contains multiple instances of "weighting", not just a single instance.

4. There is no allowance for document length. Whereas this factor is not as important as the first three factors, it can be important to normalize ranking for length because otherwise long documents often rank higher than short documents, even though the query terms may be more concentrated in the short documents.

These problems can be largely avoided by using more complex statistical ranking routines involving proper term weighting or accurate similarity measures.

Various experiments in laboratories have been concerned with developing optimal methods of weighting the terms and optimal methods of measuring the similiarity of a document and the query. One of the term weighting measures that has proven very successful is the inverted document frequency weight or IDF (Sparck Jones 1972), which is basically a measure of the scarcity of a term in the text collection. A second measure used is some function of a term's frequency within a record. These measures are often combined, with appropriate normalization factors for length, to form a single term weight. Statistically-ranked retrieval using this type of term weighting has a retrieval performance that is significantly better in the laboratory than using no term weighting (Salton & McGill 1983, Croft 1983, Harman 1986).

The following recommendations can be made based on this research.

1. The use of term weighting based on the distribution of a term within a collection usually improves performance, and never hurts performance. The IDF measure has been commonly used for this weighting.

$$IDF_i = \log_2 \frac{N}{n_i} + 1 \quad \textit{(Sparck Jones 1972)}$$

where N = the number of documents in the collection
n_i = the total frequency of term i in the collection

2. The combination of the within-document frequency with the IDF weight often provides even more improvement. It is important to normalize the within-document frequency in some manner, both to moderate the effect of high frequency terms in a document (i.e. a term appearing 20 times is not 20 times as important as one appearing only once) and to compensate for document length. Data containing very short documents (such as titles only) should not use weighting for within-document frequency. The following within-document frequency measures illustrate correct normalization procedures.

$$cfreq_{ij} = K + (1-K)\frac{freq_{ij}}{maxfreq_j} \quad \textit{(Croft 1983)}$$

$$nfreq_{ij} = \frac{\log_2 (freq_{ij}+1)}{\log_2 length_j} \quad \textit{(Harman 1986)}$$

where $freq_{ij}$ = the frequency of term i in document j
$maxfreq_j$ = the maximum frequency of any term in document j
K = the constant used to adjust for relative importance of within-document frequency
$length_j$ = the number of unique terms in document j

3. Assuming within-document term frequencies are to be used, several methods can be used for combining these with the IDF measure. Both the combining of term weighting and the use of this weighting in similarity measures between queries and documents are shown.

$$similarity(Q,D) = \frac{\sum_{i=1}^{t} (w_{iq} \times w_{ij})}{\sqrt{\sum_{i=1}^{t} (w_{iq})^2 \times \sum_{i=1}^{t} (w_{ij})^2}} \qquad \textit{(Salton \& Buckley 1988)}$$

$$\textit{where} \quad w_{iq} = (0.5 + \frac{0.5\, freq_{iq}}{maxfreq_q}) \times IDF_i$$

$$\textit{and} \quad w_{ij} = \frac{freq_{ij} \times IDF_i}{\sqrt{\sum_{vector} (freq_{ij} \times IDF_i)^2}}$$

where $freq_{iq}$ = the frequency of term i in query q
$maxfreq_q$ = the maximum frequency of any term in query q
IDF_i = the IDF of term i in the entire collection
$freq_{ij}$ = the frequency of term i in document j

Salton & Buckley suggest reducing the query weighting w_{iq} to only the within-document frequency ($freq_{iq}$) for long queries containing multiple occurrences of terms, and to use only binary weighting of documents (w_{ij} = 1 or 0) for collections with short documents or collections using controlled vocabulary.

$$similarity_j = \sum_{i=1}^{Q} (C + IDF_i \times cfreq_{ij}) \qquad \textit{(Croft 1983)}$$

$$\textit{where} \quad cfreq_{ij} = K + (1-K) \frac{freq_{ij}}{maxfreq_j}$$

where $freq_{ij}$ = the frequency of term i in document j
C = the constant used to adjust for relative importance of all term weighting
$maxfreq_j$ = the maximum frequency of any term in document j
K = the constant used to adjust for relative importance of within-document frequency

C should be set to low values (near 0) for automatically indexed collections, and to higher values such as 1 for manually-indexed collections. K should be set to low values (0.3 was used by Croft) for collections with long (35 or more terms) documents, and to higher values (0.5 or higher) for collections with short documents, reducing the role of within-document frequency.

$$similarity_j = \sum_{i=1}^{Q} \frac{\log_2 (freq_{ij}+1) \times IDF_i}{\log_2 length_j} \qquad \textit{(Harman 1986)}$$

where $freq_{ij}$ = the frequency of term i in document j
$length_j$ = the number of unique terms in document j

4. It can be very useful to add additional weight for document structure, such as higher weightings for terms appearing in the title or abstract versus those appearing only in the text. This additional weighting needs to be considered with respect to the particular text collection being used for searching.

This section on term weighting presents only a few of the experimental techniques that have been tried. For a more thorough survey, see Harman (1992a).

3.2 Query expansion

One of the problems found in all information retrieval systems is that relevant documents are missed because they contain no terms from the query. Whereas often users do not want to find most of the relevant documents, sometimes they want to find many more relevant documents and are willing to examine more documents in hopes of finding more relevant ones. However the automatic indexing systems generally do not offer the "higher-level" terms describing a document that could have been manually assigned, and it is difficult to generate a more exhaustive search. One way around this difficulty is to provide tools for query expansion. A simple example of such a tool would be the ability to browse the dictionary or word list for the text collection. Two more sophisticated techniques would be the use of relevance feedback or the use of an automatically-constructed thesaurus.

Relevance feedback is a technique that allows users to select a few relevant documents and then ask the system to use these documents to improve performance, i.e. retrieve more relevant documents. There has been a significant amount of research into using this method, although there are few user experiments on large test collections. Salton & Buckley (1990a) showed that adding relevance feedback to their similarity measure results in up to 100% improvement for small test collections. Croft (1983) used the relevant and nonrelevant documents to probabilistically change the term weighting and in 1990 he extended this work by also expanding queries using terms in the relevant documents. A similar approach was taken by Harman (1992c) and these results (again for a small test collection) showed improvements of around 100% in performance. Clearly the use of relevance judgments to improve performance is important in full-text searching and can supplement the use of the basic automatically-indexed terms, but the exact methods of using these relevance judgments is still to be determined for large full-text documents. Possibly their best use is in providing an interactive tool for modifying the query by suggesting new terms. For a survey of the use of relevance feedback in experimental retrieval systems, including Boolean systems, see Harman (1992b).

A different method of query expansion could be the use of a thesaurus. This thesaurus could be used as a browsing tool, or could be incorporated automatically in some manner. The building of such a thesaurus, however, is a massive, often domain-dependent task. Some research has been done into automatically building a thesaurus. Sparck Jones & Jackson (1970) experimented with clustering terms based on co-occurrance of these terms in documents. They tried several different clustering techniques and several different methods of using these clusters on the manually-indexed Cranfield collection. The major results on this small test collection showed that 1) it is important NOT to cluster high frequency terms (they became unit clusters), 2) it is important to create small clusters, and 3) it is better to search using the clusters alone rather than a "mixed-mode" of clusters and single terms. Crouch (1988) also generated small clusters of low freqency terms, but had good results searching using query terms augmented by thesaurus classes. Careful attention was paid to properly weighting these additional "terms". It is of course unknown how these results scale up to large full-text collections, but the concept seems promising enough to encourage further experimentation.

3.3 The use of multiple-word phrases for indexing

Large full-text collections not only need special query expansion devices to improve recall (the percentage of total relevant documents retrieved), but also need precision devices to improve their accuracy. One important precision device is the term weighting discussed in section 3.1. The ability to provide ranked output improves precision because users are no longer looking at a random ordering of selected documents. However further improvement in precision may be necessary for searching in large full-text collections, and one way to get additional accuracy is to require more stringent matching, such as phrase matching.

Phrase matching has been used in experiments in information retrieval for many years, but has currently gotten more attention because of improvements in natural language technology. The initial phrase matching used templates (Weiss 1970) rather than deep natural language parsing algorithms. The FASIT system (Dillon & Gray 1983; Burgin & Dillon 1992) used template matching by creating a dictionary of syntactic category patterns and using this dictionary to locate phrases. They assigned syntactic categories by using a suffix dictionary and exception list. The phrases detected by this system were normalized and then merged into concept groups for the final matching with queries.

A second type of phrase detection method that is based purely on statistics was investigated by Fagan (1987, 1989). This type of system relies on statistical co-occurrances of terms, as did the automatic thesaurus building described in section 3.2, but requires that these terms co-occur in more limited domains (such as within

paragraphs or within sentences), and within a set proximity of each other. Fagan investigated the use of many different parameters for selecting these phrases, and then added the phrases as supplemental index terms, i.e. all single terms were first indexed and then some additional phrases were produced.

Fagan (1987) also examined the use of complete syntactic parsing to generate phrases. The parser generated syntactic parse trees for each sentence and the phrases were then defined as subtrees of those parse trees that met certain structural criteria. Salton et. al. (1989, 1990b) compared the phrases generated for two book chapters both by the statistical methods and the syntactic methods and found that both methods generated many correct phrases, but that the overlap of those phrases was small. Salton et. al. (1990c) also tried a syntactic tagger and bracketer (Church 1988) to identify phrases. The tagger uses statistical methods to produce syntactic part-of-speech tags, and the bracketer identifies phrases consisting of noun and adjective sequences. This simpler approach does not require the completion of entire parse trees and seemed to produce as many good phrases.

In general, retrieval experiments that add phrases to single term indexing have not been successful with small test collections. One reason has been the scarcity of phrases in the text that match phrases in the query. Lewis & Croft (1990) tried first locating phrases using a chart parser, and then clustering these phrases. The retrieval used single terms, phrases, and clustered phrases in different combinations. The best performance used terms, phrases, and clustered phrases as features for retrieval. However even this performance was not significantly better than performance using only single terms for the small test collection used.

The current feeling among researchers is that the use of multiple-word phrases will be successful only for large collections of text. This is partially because of the need for enough text to locate phrases that will be good features for retrieval. Equally important, the higher precision retrieval offered by phrases may only be important in the larger full-text retrieval environment. Croft et. al. (1991) investigated various ways of both generating and using phrases in retrieval, and although their results on the small CACM test collection were not significant, the work they are doing on a larger test collection shows impressive results using phrases. It is likely that the use of phrases for retrieval in large full-text retrieval environments will show significant, and possibly critical, improvements over single term indexing.

3.4 Feature selection

Another method of improving precision in retrieval from large full-text data is to select indexing features more carefully. The current approach to automatic indexing generally indexes all stems in a document, eliminating only stopwords and possibly numbers. This exhaustive coverage may be important for small documents such as abstracts or bibliographic records, but using all terms in very large records may weaken the matching criteria. Ideally one would like to be able to automatically select the single terms or phrases which best represent a document. Unfortunately this area has attracted little research because of the absence of large full-text test collections.

Two recent papers address this issue. The first paper (Strzalkowski 1992) described some research using a statistical retrieval system with some improvements based on natural language techniques. Strzalkowski used a very fast syntactic parser to parse the text. The phrases found using this parser were then statistically analyzed and filtered to produce automatically a set of semantic relationships between the words and subphrases. This highly selective set of phrases was then used to both expand and filter the query. The results on the small CACM collection showed a significant improvement in performance over the straight statistical methods, and these techniques clearly will scale up to larger full-text documents.

The second paper (Lewis 1992) was an investigation into feature selection using a classification test collection. This test collection contains 21,450 Reuters newswires that have been manually classified into 135 topic descriptions. The goal of this research was to identify what text features (terms, phrases, or phrase clusters) were important in generating these categories. Best results were obtained for a small number of features (10 - 15), and some discussion is made of the best ways to select these features. This type of approach to feature "pruning" also needs to be further explored for large full-text collections.

3.5 More advanced term weighting techniques

A third approach to increased precision for the larger documents is to use all terms for indexing, but to provide more sophisticated term weighting methods than those discussed in section 3.1. Salton & Buckley (1991)

presented results from work using an online encyclopedia in which they weighted terms both globally for an entire document (as in section 3.1), but also locally for a given sentence. In this particular experiment they performed multiple-stage searching in which a short initial query was used to find one or more relevant sections or paragraphs, and then these sections were used to find similar sections using both global and local weighting schemes. Whereas the global weights help increase the recall by returning many similar items, the local weights can be used as a filtering operation to improve the precision of the returned set. Further details can be found in a technical report (Salton & Buckley 1990d). This type of approach to searching and term weighting may be particularly suitable for large full-text data collections.

3.6 Using combinations of indexing techniques

All the preceding research efforts had as a basis the combination of various information from the text to improve indexing and searching. The best term weighting schemes discussed in section 3.1 combined different statistical measures of term importance. The section on query expansion dealt with combining information about term co-occurrence to automatically identify better query terms and term weights. The work on multiple word phrases investigated how to locate phrases, but also how to correctly combine these phrases with single terms. Feature selection involves combining information from the text to help better select which features to index, and the advanced term weighting techniques combine term weights at two granularity levels to improve precision.

Other more explicit combination techniques have been tried, from simple user weighting of terms (to be combined with the statistical term weighting), to combining of database attributes with free text (Deogun & Raghavan 1988), to more elaborate combining of concepts such as citations, attributes, and data into the vector space model (Fox et. al. 1988). Results have generally shown improvements in performance, even for small test collections. This combination of various sources of information can be extended to combining various types of indexing (such as manual or automatic), various types of queries (such as using or not using Boolean connectors), or various types of searching (such as cluster searching vs document searching). It has been shown (Katzer et. al. 1982) that different indexing or searching methods can produce comparable results, but with *little* overlap between the sets of relevant documents. Clearly it would be ideal to combine these methods, but the method for combining the completely different approaches to indexing and searching is not easily apparent.

A new model, the inference network (Turtle and Croft 1991) is designed specifically for this task of combining evidence or probabilities from all these different methods. This network consists of term nodes, document nodes, and query nodes, connected by links with probabilitistic weighting factors, and can be used to try multiple ways of combining information from these nodes to form a list of documents ranked in order of likely relevance to a user's need. Turtle & Croft show how this model can be used to represent most of the basic indexing and searching techniques, and discuss how the generation of this model provides the scope for a thorough investions of how to perform complex combinations of techniques. This type of representation can be viewed as a very advanced indexing method, and may prove important in handling large full-text data.

3.7 Summary

Whereas the traditional automatic single term indexing described in section 2 enables reasonable searching of large full-text documents, the more advanced techniques discussed in this section may all prove important in raising the retrieval performance beyond a mediocre level. It is critical that research continue into these advanced techniques, and others like them, and that as they become proven methodologies, they be accepted as standard automatic indexing techniques by the information retrieval community as a whole.

REFERENCES

Burgin R. and Dillon M. (1992). Improving Disambiguation in FASIT. *Journal of the American Society for Information Science*, 43(2), 101-114.

Church K. (1988). A Stochastic Part Program and Noun Phrase Parser for Unrestricted Text. In: Proceedings of the Second Conference on Applied Natural Language Processing; 1988, 136-143; Austin, Texas.

Cleverdon C.W. and Keen E.M. (1966). *Factors Determining the Performance of Indexing Systems, Vol. 1:*

Design, Vol. 2: Test Results. Aslib Cranfield Research Project, Cranfield, England, 1966.

Croft W.B. (1983). Experiments with Representation in a Document Retrieval System. *Information Technology: Research and Development*, 2(1), 1-21.

Croft W.B. and Das R.(1990). Experiments with Query Acquistion and Use in Document Retrieval Systems. In: Proceedings of the 13th International Conference on Research and Development in Information Retrieval; September 1990, 349-368; Brussels, Belgium.

Croft W.B., Turtle H., and Lewis D.(1991). The Use of Phrases and Structured Queries in Information Retrieval. In: Proceedings of the 14th International Conference on Research and Development in Information Retrieval; October 1991, 32-45; Chicago, Illinois.

Crouch C.J. (1988). A Cluster-Based Approach to Thesaurus Construction. In: Proceedings of the ACM Conference on Research and Development in Information Retrieval; June 1988, 309-320; Grenoble, France.

Deogun J.S. and Raghavan V.V. (1988). Integration of Information Retrieval and Database Management Systems. *Information Processing and Management*, 24(3), 303-313.

Dillon M. and Gray A.S. (1983). FASIT: A fully automatic syntactically based indexing system. *Journal of the American Society for Information Science*, 34(2), 99-108.

Fagan J. (1987). *Experiments in Automatic Phrase Indexing for Document Retrieval: A Comparison on Syntactic and Nonsyntactic Methods.* Doctoral dissertation, Cornell University, Ithaca, N.Y.

Fagan J. (1989). The Effectiveness of a Nonsyntactic Approach to Automatic Phrase Indexing for Document Retrieval, *Journal of the American Society for Information Science*, 40(2), 115-132.

Fox C. (1992). Lexical Analysis and Stoplists. In Frakes W.B. and Baeza-Yates R., (Ed.), *Information Retrieval: Data Structures and Algorithms*, Englewood Cliffs, NJ.: Prentice-Hall.

Fox E.A., Nunn G.L., and Lee W.C. (1988). Coefficients of Combining Concept Classes in a Collection. In: Proceedings of the ACM Conference on Research and Development in Information Retrieval; June 1988, 291-308; Grenoble, France.

Frakes W.B. (1984). Term Conflation for Information Retrieval. In: Proceedings of the Third Joint BCS and ACM symposium on Research and Development in Information Retrieval; July 1984, 383-390; Cambridge, England.

Frakes W.B. (1992). Stemming Algorithms. In Frakes W.B. and Baeza-Yates R., (Ed.), *Information Retrieval: Data Structures and Algorithms*, Englewood Cliffs, NJ.: Prentice-Hall.

Francis W. and Kucera H. (1982). *Frequency Analysis of English Usage*, New York, NY.: Houghton Miffin.

Harman D. (1986). An Experimental Study of Factors Important in Document Ranking. In: Proceedings of the ACM Conference on Research and Development in Information Retrieval; September 1986, 186-193; Pisa, Italy.

Harman D. (1991). How Effective is Suffixing? *Journal of the American Society for Information Science*, 42(1), 7-15.

Harman D. (1992a). Ranking Algorithms. In Frakes W.B. and Baeza-Yates R., (Ed.), *Information Retrieval: Data Structures and Algorithms*, Englewood Cliffs, NJ.: Prentice-Hall.

Harman D. (1992b). Relevance Feedback and Other Query Modification Techniques. In Frakes W.B. and Baeza-Yates R., (Ed.), *Information Retrieval: Data Structures and Algorithms*, Englewood Cliffs, NJ.: Prentice-Hall.

Harman D.(1992c). Relevance Feedback Revisited. In: Proceedings of the 15th International Conference on Research and Development in Information Retrieval; June 1991, 1-10; Copenhagen, Denmark.

Harman D. and Candela G. (1990). Retrieving Records from a Gigabyte of Text on a Minicomputer using Statistical Ranking, *Journal of the American Society for Information Science*, 41(8), 581-589.

Katzer J., McGill M.J., Tessier J.A., Frakes W., and DasGupta P. (1982). A Study of the Overlap among Document Representations. *Information Technology: Research and Development*, 1(2), 261-274.

Lewis D.D. (1992). Feature Selection and Feature Extraction for Text Categorization. Paper to appear in the Proceedings of the 5th DARPA Workshop on Speech and Natural Language, Harriman, N.Y.

Lewis D.D. and Croft W.B. (1990). Term Clustering of Syntactic Phrases, In: Proceedings of the 13th International Conference on Research and Development in Information Retrieval; September 1990; 385-504; Brussels, Belgium.

Lennon M., Peirce D., Tarry B., Willett P. (1981). An Evaluation of Some Conflation Algorithms for Information Retrieval, *Journal of Information Science*, 3, 177-188.

Lovins J.B. (1968). Development of a Stemming Algorithm, *Mechanical Translation and Computational Linguistics*, 11, 22-31.

Luhn H.P. (1957). A Statistical Approach to Mechanized Encoding and Searching of Literary Information, *IBM Journal of Research and Development*, 1(4), 309-317.

Pacak M.G. and Pratt A.W. (1978). Identification and Transformation of Terminal Morphemes in Medical English, Part II, *Methods of Information in Medicine*, 17, 95-100.

Porter M.F. (1980). An Algorithm for Suffix Stripping, *Program*, 14(3),130-137.

Salton G. and Buckley C. (1988). Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*, 24(5), 513-523.

Salton G., and Buckley C. (1989). A Comparison between Statistically and Syntactically Generated Term Phrases. *Technical Report TR 89-1027*, Cornell University: Computing Science Department.

Salton G. and Buckley C. (1990a). Improving Retrieval Performance by Relevance Feedback. *Journal of the American Society for Information Science*, 41(4), 288-297.

Salton G., and Buckley C. (1990d). An Evaluation of Text Matching Systems for Text Excerpts of Varying Scope. *Technical Report TR 89-1027*, Cornell University: Computing Science Department.

Salton G. and Buckley C. (1991). Automatic Text Structuring and Retrieval: Experiments in Automatic Encyclopedia Searching. In: Proceedings of the 14th International Conference on Research and Development in Information Retrieval; October 1991, 21-31; Chicago, Illinois.

Salton G., Buckley C., and Smith M. (1990b) On the Application of Syntactic Methodologies in Automatic Text Analysis, *Information Processing and Management*, 26(1), 73-92.

Salton G., Zhao Z. and Buckley C. (1990c). A Simple Syntactic Approach for the Generation of Indexing Phrases *Technical Report TR 90-1137*, Cornell University: Computing Science Department.

Salton G. and McGill M. (1983). *Introduction to Modern Information Retrieval*. New York, NY.: McGraw-Hill.

Sparck Jones K. (1972). A Statistical Interpretation of Term Specificity and Its Application in Retrieval. *Journal of Documentation*, 28(1), 11-20.

Sparck Jones K. and Jackson D.M. (1970). The Use of Automatically-Obtained Keyword Classifications for Information Retrieval. *Information Storage and Retrieval*, 5, 175-201.

Strzalkowski T. (1992). Information Retrieval using Robust Natural Language. Paper to appear in the Proceedings of the 5th DARPA Workshop on Speech and Natural Language, Harriman, N.Y.

Turtle H. and Croft W.B. (1991). Evaluation of an Inference Network-Based Retrieval Model. *ACM Transactions on Information Systems*, 9(3), 187-222.

van Rijsbergen C.J. (1975). *Information Retrieval*. London: Butterworths.

Walker, S. and Jones, R. M. (1987). *Improving Subject Retrieval in Online Catalogues*, British Library Research Paper 24.

Weiss S.F. (1970). *A Template Approach to Natural Language Analysis for Information Retrieval*. Doctoral dissertation, Cornell University, Ithaca, N.Y.

NIST-114A
(REV. 3-90)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

BIBLIOGRAPHIC DATA SHEET

1. PUBLICATION OR REPORT NUMBER
NISTIR 4873

2. PERFORMING ORGANIZATION REPORT NUMBER

3. PUBLICATION DATE
SEPTEMBER 1992

4. TITLE AND SUBTITLE

Automatic Indexing

5. AUTHOR(S)

Donna Harman

6. PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
GAITHERSBURG, MD 20899

7. CONTRACT/GRANT NUMBER

8. TYPE OF REPORT AND PERIOD COVERED

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)

10. SUPPLEMENTARY NOTES

11. ABSTRACT (A 200-WORD OR LESS FACTUAL SUMMARY OF MOST SIGNIFICANT INFORMATION. IF DOCUMENT INCLUDES A SIGNIFICANT BIBLIOGRAPHY OR LITERATURE SURVEY, MENTION IT HERE.)

Automatic indexing has been a critical technology as more full-text data becomes available online. This paper discusses issues for automatic indexing of different types of full-text and also presents a survey of much of the current research into new techniques for automatic indexing.

12. KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)

automatic indexing; full-text; information retrieval

13. AVAILABILITY

X	UNLIMITED
	FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS).
	ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402.
X	ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161.

14. NUMBER OF PRINTED PAGES
17

15. PRICE
A02

ELECTRONIC FORM